

Aim:

Write Java program(s) on creating multiple threads, assigning priority to threads, synchronizing threads, suspend and resume threads

Source Code:**TestThread.java**

```
class RunnableDemo implements Runnable {
    public Thread t;
    private String threadName;
    boolean suspended = false;
    RunnableDemo(String name) {
        threadName = name;
        System.out.println("Creating " + threadName);
    }
    public void run() {
        System.out.println("Running " + threadName);
        try {
            for (int i = 10; i > 0; i--) {
                System.out.println("Thread: " + threadName + ", " + i);
                Thread.sleep(200);
                synchronized(this) {
                    while (suspended) {
                        wait();
                    }
                }
            }
        } catch (InterruptedException e) {
            System.out.println("Thread " + threadName + " interrupted.");
        }
        System.out.println("Thread " + threadName + " exiting.");
    }
    public void start() {
        System.out.println("Starting " + threadName);
        if (t == null) {
            t = new Thread(this, threadName);
            t.start();
        }
    }
    void suspend() {
        suspended = true;
    }
    synchronized void resume() {
        suspended = false;
        notify();
    }
}

public class TestThread {
    public static void main(String args[]) {
        RunnableDemo R1 = new RunnableDemo("Thread-1");
        R1.start();
    }
}
```

```

RunnableDemo R2 = new RunnableDemo("Thread-2");
R2.start();
try {
    Thread.sleep(300);
    R1.suspend();
    System.out.println("Suspending First Thread");
    Thread.sleep(300);
    R1.resume();
    System.out.println("Resuming First Thread");
    R2.suspend();
    System.out.println("Suspending thread Two");
    Thread.sleep(300);
    R2.resume();
    System.out.println("Resuming thread Two");
} catch (InterruptedException e) {
    System.out.println("Main thread Interrupted");
}
try {
    System.out.println("Waiting for threads to finish.");
    R1.t.join();
    R2.t.join();
} catch (InterruptedException e) {
    System.out.println("Main thread Interrupted");
}
System.out.println("Main thread exiting.");
}
}

```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Creating Thread-1
Starting Thread-1
Creating Thread-2
Starting Thread-2
Running Thread-1
Running Thread-2
Thread: Thread-2, 10
Thread: Thread-1, 10
Suspending First Thread
Thread: Thread-2, 9
Thread: Thread-2, 8
Resuming First Thread
Suspending thread Two
Thread: Thread-1, 9
Thread: Thread-1, 8
Resuming thread Two
Waiting for threads to finish.
Thread: Thread-2, 7
Thread: Thread-1, 7
Thread: Thread-2, 6
Thread: Thread-1, 6
Thread: Thread-2, 5

Thread: Thread-1, 5
Thread: Thread-2, 4
Thread: Thread-1, 4
Thread: Thread-2, 3
Thread: Thread-1, 3
Thread: Thread-2, 2
Thread: Thread-1, 2
Thread: Thread-2, 1
Thread: Thread-1, 1
Thread Thread-2 exiting.
Thread Thread-1 exiting.
Main thread exiting.